RCA *PD040010AC*

CITED BY APPLICANT
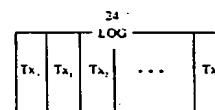
(54) INDEXATION CONDITIONNEL DE DONNEES A BASE HEURISTIQUE
(54) HEURISTIC-BASED CONDITIONAL DATA INDEXING

(57)
A computer system for the indexing of data in which a heuristic determination function is applied to predict an efficient index updating approach. The system is able to update an index relating to a first data set by incrementally updating the index or by a rebuild of the index at the completion of the addition of a second set of data to the first set of data. The system applies a heuristic determination function to the characteristics of the first set of data, its index, and the second set of data, to predict whether an incremental update or a rebuild update of the index will result in a more efficient rebuild of the data. The system applies this approach to a restore and rollforward recovery or a data load operation to improve the efficiency of these operations.

(72) LIGHTSTONE, SAM S., CA
(72) FLASZA, MIROSLAW A., CA
(72) McARTHUR, CATHERINE S., CA
(72) O'CONNELL, WILLIAM T., CA
(71) IBM CANADA LTD. - IBM CANADA LIMITÉE, CA

(51) Int.Cl.⁶ G06F 17/30

 (51) Int.Cl.^{6} G06F 17/30
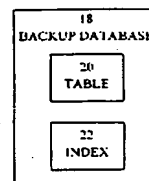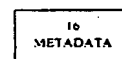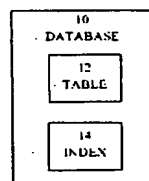
(54) INDEXATION CONDITIONNEL DE DONNEES A BASE
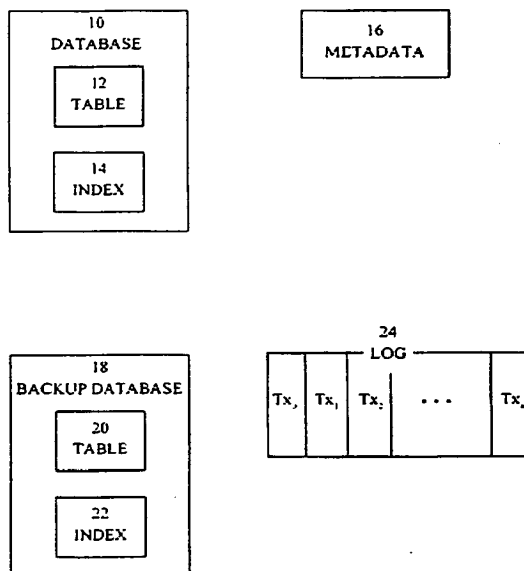HEURISTIQUE
(54) HEURISTIC-BASED CONDITIONAL DATA INDEXING

(57) A computer system for the indexing of data in which a heuristic determination function is applied to predict an efficient index updating approach. The system is able to update an index relating to a first data set by incrementally updating the index or by a rebuild of the index at the completion of the addition of a second set of data to the first set of data. The system applies a heuristic determination function to the characteristics of the first set of data, its index, and the second set of data, to predict whether an incremental update or a rebuild update of the index will result in a more efficient rebuild of the data. The system applies this approach to a restore and rollforward recovery or a data load operation to improve the efficiency of these operations.

CA9-99-022

HEURISTIC-BASED CONDITIONAL DATA INDEXING

ABSTRACT

5   A computer system for the indexing of data in which a heuristic determination function is applied
to predict an efficient index updating approach. The system is able to update an index relating to
a first data set by incrementally updating the index or by a rebuild of the index at the completion of
the addition of a second set of data to the first set of data. The system applies a heuristic
determination function to the characteristics of the first set of data, its index, and the second set of
10   data, to predict whether an incremental update or a rebuild update of the index will result in a more
efficient rebuild of the data. The system applies this approach to a restore and rollforward recovery
or a data load operation to improve the efficiency of these operations.

CA9-99-022

HEURISTIC-BASED CONDITIONAL DATA INDEXING

FIELD OF THE INVENTION

The present invention is directed to an improvement in computing systems and in particular to
5   computer systems which include the indexing of data.

BACKGROUND OF THE INVENTION

Computer systems for the storage and retrieval of data typically include indexes of the data which
are created to permit operations on the data (data retrieval or updating) to be performed in an
10   efficient manner. Updating such indexes is typically costly in system resources and time. For
example, in a relational database environment, appending a new record to a table in the database will
require a new entry in any indexes which have been created for that table. The time and computer
resources required to add a set of data to the table may be significantly less than the operation of
updating the associated indexes.

15

In database management system (DBMS) environments, it is typically the case that the efficiency
or inefficiency of the index updating operations significantly impact on the efficiency of the database
operations as a whole.

20   It is therefore desirable to have a DBMS computer system which includes features to make more
efficient the updating of indexes associated with data stored by the DBMS.

SUMMARY OF THE INVENTION

According to one aspect of the present invention, there is provided an improved system for indexing
25   data.

According to another aspect of the present invention, there is provided a computer system for

1

CA9-99-022

database management comprising, means for storing and updating a first set of data, indexing means for storing and updating a selected index of key values related to the first set of data, the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, heuristic determination means for selecting the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data.

5

According to another aspect of the present invention, there is provided the above computer system in which the heuristic determination means comprises a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

10

According to another aspect of the present invention, there is provided the above computer system in which the database management system is a relational database management system in which the index is stored as a binary tree and in which the index meta-data comprises estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index.

15

According to another aspect of the present invention, there is provided the above computer system in which the function determines, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

20

25

According to another aspect of the present invention, there is provided the above computer system in which the heuristic determination means comprises means for the user to specify the selection of

CA9-99-022

the incremental update of the index or the full rebuild of the index.

According to another aspect of the present invention, there is provided the above computer system
further comprising means for generating values for the function of the heuristic determination means,

5   for a specified test range of values for each of the first set of data, the index, and the second set of
data.

According to another aspect of the present invention, there is provided the above computer system,
the computer system having one or more CPUs, one or more disks, a sort heap and a database

10   bufferpool, and in which the index meta-data is defined to reflect a subset of the following
characteristics: the percentage of free space in the index, the estimated size of the index after both
the incremental and the rebuild updates of the index, the width of the average key value in the index,
the size of the sort heap and the database bufferpool in the computer system, the number and speed
of the CPUs in the computer system, and the number and speed of the disks in the system.

15

According to another aspect of the present invention, there is provided a computer system for
relational database management comprising, means for storing and updating a first set of data,
indexing means for storing as a binary tree, and updating, a selected index of key values related to
the first set of data the indexing means comprising means for selectively updating the index by

20   incrementally updating the index by incrementally adding key values to the index or by fully
rebuilding the index, heuristic determination means for selecting the incremental update of the index,
or the full rebuild of the index, for a given second set of data to be added to the first set of data, the
heuristic determination means comprising a function which takes as input index meta-data, the index
meta-data being stored separately from the first set of data and comprising estimates of the table size

25   of the first set of data, the table size of the second set of data, and the height of the index, whereby
the function is empirically defined to select the incremental update of the index or the full rebuild
of the index based on the predicted relative efficiencies of the incremental update of the index and

3

CA9-99-022

the full rebuild of the index, the heuristic determination means further comprising means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

According to another aspect of the present invention, there is provided a computer program product
5   for use with a computer comprising a central processing unit and random access memory, said computer program product comprising a computer usable medium having computer readable code means embodied in said medium for managing a database, said computer program product comprising computer readable program code means for causing a computer to implement the system functionality described above.
10

According to another aspect of the present invention, there is provided a method for updating a selected index in a computer system for database management, the computer system comprising means for storing and updating a first set of data related to the index, the method comprising the steps of:
15   (a) heuristically selecting one of: the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, and
(b) selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, as selected.

20   According to another aspect of the present invention, there is provided the above method in which the step of heuristically selecting the update of the index comprises the evaluation of a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

25   According to another aspect of the present invention, there is provided a method for updating a selected b-tree index in a computer system for relational database management, the computer system comprising means for storing and updating a first set of data related to the index, the method

4

CA9-99-022

comprising the steps of:

(a) using a heuristic determination function to select one of: the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, the heuristic determination function taking as input index meta-data comprising estimates of the table

5   size of the first set of data, the table size of the second set of data, and the height of the index, whereby the heuristic function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index, and

(b) selectively updating the index by incrementally updating the index by incrementally adding key

10   values to the index or by fully rebuilding the index, as selected.

According to another aspect of the present invention, there is provided the above method in which the function comprises the step of determining, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data,

15   whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

According to another aspect of the present invention, there is provided the above method further comprising the step of a user specifying the selection of the incremental update of the index, the full

20   rebuild of the index, or the selection based on the heuristic determination function.

According to another aspect of the present invention, there is provided a computer program product tangibly embodying a program of instructions executable by a computer to perform the above method steps.

25   Advantages of the present invention include improvements in the efficiency in updating a database when loading data or when rolling forward data after a database restore.

5

CA9-99-022

BRIEF DESCRIPTION OF THE DRAWINGS

The preferred embodiment of the invention is shown in the drawings, wherein:

Figure 1 is a block diagram showing the data structures used by the preferred embodiment in a data roll forward.

Figure 2 is a block diagram showing the data structures used by the preferred embodiment prior to data being loaded into a database in the preferred embodiment.

Figure 3 is a block diagram showing the data structures used by the preferred embodiment during data being loaded into a database in the preferred embodiment.

In the drawings, the preferred embodiment of the invention is illustrated by way of example. It is to be expressly understood that the description and drawings are only for the purpose of illustration and as an aid to understanding, and are not intended as a definition of the limits of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The preferred embodiment is described with reference to the DB2™ database management system (DBMS). Those skilled in the art will appreciate that the preferred embodiment may be implemented in other DBMS environments other than DB2™. The preferred embodiment of the invention relies on the fact that for some database operations a series of transactions will result in data being updated without there being a need to access indexes associated with the data.

For example, a DBMS may permit a user to load data into a table in the database from another source (for example where data is copied from a first database into a second database). The DB2™ LOAD command is an example of this sort of operation. The LOAD command will result in data being inserted into a table. It is possible for large data collections to be added to tables using the LOAD command. In one type of prior art system, the DBMS will incrementally update indexes associated with the table into which data is placed using the LOAD command. In such a system, the index on the table will be modified on a record by record basis to reflect the new records added to the table by the LOAD command. However, the index will not be used, in the sense of being

6

CA9-99-022

accessed to locate data in the database, until the LOAD command has been completed and all the data has been added to the target table in the database. In other systems, the index will not be updated in an incremental fashion but will be fully rebuilt following the addition of new records to the data in the database.

5

Another example of where a DBMS adds to or modifies data without the use of an index until the changes to the data are complete is in the ROLLFORWARD command in DB2™. When data in a relational database is recovered by using the RESTORE and ROLLFORWARD commands, the DBMS facilitates the rebuilding of a database from a transaction log where the database has been

10 destroyed or become corrupted in some manner (a database "roll forward"). During the roll forward process, the data in the database is modified to reflect the transactions stored in the transaction log. In the prior art, it is common to update associated indexes on a record by record basis which reflects the transactions stored in the transaction log. For DBMS environments such as the DB2 ™ system, the transaction log stores a portion of the transactional history which permits the updating of the data

15 in the database after a database restore, without requiring reference to the indexes associated with the data being updated or rolled forward. The indexes associated with the data must be brought up to date at the end of the roll forward operation, but need not be synchronized with the data during the operation.

20 As indicated above, for certain DBMS environments, the DBMS will update indexes on a transaction by transaction or record by record basis. For certain DBMS environments, the content of the indexes during the data updating process (during the LOAD or the roll forward command execution) is not accessed by the DBMS. The DBMS requires a correct set of indexes at the completion of the operations but is not concerned with the intermediate values of the indexes.

25

As the above indicates, in the prior art, one approach in loading data or performing a roll forward on a database, is to update the database on a record by record basis. In such a system any indexes

7

CA9-99-022

which are associated with the data being updated will also be modified on a record by record basis. Alternatively, certain DBMS designs will permit indexes to be rebuilt at the end of a roll forward operation. In certain DBMS designs, it is also possible for the system itself to select an index rebuild in certain circumstances where otherwise a record by record or incremental update of the index

5   would occur. In the DB2™ DBMS, for example, the default approach during the ROLLFORWARD command execution is the incremental updating of the index. The index rebuilding operation is only carried out where one of the transactions being replayed in a roll forward is an index create function. Otherwise, the entire set of transactions is replayed on the index.

10   Where data is loaded or a database is subject to a roll forward operation, it is possible that a significant proportion of the processing time for the operation will be taken up by the addition to or modification of the indexes associated with the data. In fact, for certain databases it may be more efficient to ignore the incremental updating of the associated indexes and to fully rebuild the indexes at the completion of the loading of data or the roll forward operations.

15

Turning to Figure 1, an example database 10 is illustrated. Database 10 contains table 12 and index 14. For the purposes of the description of the preferred embodiment a relational database is described. It will be apparent to those skilled in the art that other database systems may be used to implement the preferred embodiment, with the appropriate changes to accommodate differences

20   from the relational database model. In the preferred embodiment, there is a collection of meta-data shown in Figure 1 as meta-data 16.

In the preferred embodiment, a backup of the database is made at periodic or user-defined intervals. In Figure 1, backup database 18 is shown with table 20 and index 22. Backup database 18 represents

25   stored data for database 10. In the system of the preferred embodiment, there is also a set of transactions stored in log 24. The stored transactions are shown as the set Tx_0, Tx_1, Tx_2,...Tx_n. In the preferred embodiment, the set of transactions stored in log 24 are used by the system in

8

CA9-99-022

conjunction with backup database 18 to recover data which may have been lost or inadvertently deleted. In the system of the preferred embodiment, the database 10 may be replaced with the data from backup database 18 (using the RESTORE command in the DB2™ DBMS, for example). Log 24 contains transactions which have occurred since the time that backup database 18 was created.

5   The stored transactions in log 24 are able to be replayed on recovered data in database 10, to update table 12 and index 14 in database 10. One prior art approach is to update index 14 as each transaction in log 24 is replayed on the data in database 10. Index 14 is therefore updated on a record by record (or transaction by transaction, basis). This is the approach used by the roll forward command available in the DB2™ DBMS.

10   According to the preferred embodiment, the DBMS includes functionality to recover data using commands such as RESTORE and ROLLFOWARD, which does not necessarily require the record by record updating of index 14. In the preferred embodiment, backup database 18 may be used to replace the table 12 in database 10. However, when the transactions in log 24 are replayed on

15   database 10 as in a roll forward, the system of the preferred embodiment will selectively permit table 12 to be updated, without updating index 14 (as is set out above, log 24 contains data for each of the stored transactions to permit updating of table 12 without reference to index 14). As the time for incrementally updating index 14 is potentially long, the savings in updating table 12, only are potentially significant. Once table 12 has been updated by the transactions in log 24, index 14 will

20   be recreated by being rebuilt on the newly updated table 12.

Such a method of updating by ignoring index 14 until the completion of updating table 12 (the full index rebuild) is available in the preferred embodiment by user selection. Alternatively, the full index rebuild roll forward is available by the system of the preferred embodiment applying a

25   heuristic determination function to meta-data 16.

Index meta-data 16 is represented as S_i. A heuristic determination function $F(S_i)$ is defined in the

9

CA9-99-022

system of the preferred embodiment to accept the set of heuristic information and to determine whether indexing in a roll forward should be maintained transactionally (incremental updating), or deferred for a single recreation pass (full index rebuild).

5   The determination function $F(S_i)$ may accept as parameters such database meta-data as the estimated number of key-parts assumed to comprise an index at the time roll forward processing starts, and an estimate of the number of keypart updates that will be applied during roll forward processing. The specific decision making logic of the heuristic determination function $F(S_i)$ may be dependent on implementation details, for specific DBMS systems. The $F(S_i)$ definition may have considerable
10   empirical dependence, which may vary between products, and between product releases.

A simple implementation of function $F(S_i)$ has parameters reflecting characteristics of an index 22, and compares these to a set of index characteristics at roll forward completion time which have been maintained in meta-data 16. In the specific implementation for the DB2™ Universal Database, the
15   preferred embodiment stores the number of index key operations as meta-data inside the index object meta-index. This information is periodically reflected in the database recovery history file during normal transaction processing of the DBMS. In the preferred embodiment this information is stored outside the database to maintain the information should the database crash. During roll forward recovery processing the desired heuristic information reflecting end-of-logs can be retrieved from
20   the Recovery History File, and provided to the heuristic determination function $F(S_i)$.

In the system of the preferred embodiment, the heuristic determination function $F(S_i)$ is used also when pre-existing data is loaded into a database. Figure 2 shows database 10 with table 12 and index 14. Data 26 is data to be added to database 10. In DBMS systems such as DB2™, commands for
25   loading data such as data 26 into database 10 are typically available (the LOAD command in DB2™). In the system of the preferred embodiment, the data may be loaded such that index 14 is incrementally updated, or the data may be added to table 12 in database 10 without incrementally

10

CA9-99-022

updating index 14. At the end of adding data 26 to table 12, index 14 may then be recreated using the updated data in table 12.

The implementation of the loading of data in the preferred embodiment system may be described
5   with reference to Figure 3. The first step in loading data 26 is the addition of the data to table 12. The added data 38 shown in table 12 in Figure 3, is added from data 26 (shown in Figure 2). The system of the preferred embodiment also creates a data structure to hold the sorted keys 30 for added data 38. The sorted keys 30 relate to the keys in index 14. If the user or the heuristic determination function F(Si) selects an incremental update of index 14, then sorted keys 30 are added to index 14,
10   on a key value by key value basis. Alternatively, if the user or function $F(S_i)$ determines that index 14 should be recreated, then index 14 will be rebuilt on the entire table 12 key values.

As is apparent, in both the data recovery and roll forward operations, and in the data load operation, the use of the heuristic determination function $F(S_i)$ is available in the system of the preferred
15   embodiment. The function will be used to select incremental updating index 14, or alternatively to select a full rebuilding of index 14. In the system of the preferred embodiment, the user may either manually select incremental or full rebuilding modes or may select a system option to have the system itself use the heuristic determination function $F(S_i)$ to make the selection.

20   In both the load and restore/roll forward operations, the heuristic determination function $F(S_i)$ is intended to predict which of the two index updating approaches (incremental and full rebuild) will be most efficient for the database updating which is being performed. The case where database 10 is relatively large, and few data values are added to database 10 (and correspondingly to index 14), will be, for many DBMS environments, one where incremental updating of index 14 will be more
25   efficient. Conversely, where database 10 is relatively small and the data to be added by either a large log 24 or a large set of data 26, in many DBMS environments, it will be more efficient to update index 14 by fully rebuilding the index after table 12 has been updated.

11

CA9-99-022

Typically, it is the database subsystem and environment which will determine which of the two different updating methods is more efficient, for a given data and index updating operation. The heuristic determination function $F(S_i)$ is intended to predict which of the two methods is more efficient, given a particular combination of existing database, update data, database subsystem and

5   environment. Once the appropriate values for the function are set for a given database subsystem and environment, it is intended that the same function $F(S_i)$ will be appropriate for different table and index combinations in different database definitions.

In the system of the preferred embodiment, index 14 is stored in the form of a binary tree, such as

10   a B-tree, B+-tree, or B*-tree. It is to be understood, however, that other index structures may be used and the benefit of the invention achieved. The preferred embodiment uses two characteristics of a binary tree index to model the efficiency of incremental updating and full rebuilding for an index. The two characteristics are the percentage of new data that is to be added to table 12 and the height of the binary tree structure of index 14. The tree height of index 14 is a measure of the size

15   and complexity of index 14. In the example of the preferred embodiment, there is a single index shown. In the system of the preferred embodiment, the first table non-meta-index table index is taken to be representative of the database indexes which will be affected by the updating procedure. It will be understood by those skilled in the art that there may be other approaches to selecting an index which is representative of the indexes in a database for which data is to be updated. The

20   approach of selecting the first such index and assuming that that index is representative has the advantage of carrying low storage and processing overheads. It may be advantageous in certain circumstances to maintain other data regarding the indexes such as the largest index, the smallest index, or the median or average index. Such approaches may require more system resources to maintain but will have the advantage of potentially providing better information about the structure

25   of the indexes for the determination heuristic $F(S_i)$.

In defining $F(S_i)$ in terms of tree height of index 14, and the percentage of new data to be added to

12

CA9-99-022

table 12, $F(S_i)$ is designed to reflect the time required to extract index keys from table 12. This operation is necessary in a full rebuild of index 14 and requires a scan of all records in table 12. The cost of this is generally linear with the size of the data on disk. Typically, each page must be scanned from disk, then scanned for records, and each record scanned for index keys.

5

The function $F(S_i)$ is also designed to reflect the time to sort index keys. This varies with the sorting algorithm used, the size of buffering for sort operations, and the cost of spilling to disk (if required). Generally, however, this cost is modelled as an NlogN operation.

10   The function is also intended to reflect the time to fully rebuild index 14 from a set of sorted keys. This is generally a linear operation, which varies with the number of key values to be included in index 14. Conversely, the function will be used to reflect the time to incrementally extend index 14 with a set of sorted keys. This is generally a non-linear operation, which is a function of the height of the binary index tree.

15

In the model for $F(S_i)$ of the preferred embodiment, a threshold value for percentage of new data is determined for different tree heights. The threshold value will indicate at what point it is estimated that incremental updating of index 14 is to be used (the threshold value and below) and at what point a full rebuild approach to index 14 should be used (above the threshold value).

20

In the preferred embodiment, the $F(S_i)$ is essentially a table:

25

13

CA9-99-022

Tree Height	Threshold
2 (or less)	T2
3	T3
4	T4
5	T5
6	T6
7	T7
8	T8

In the table $T2...T8$ represent the percentile thresholds for the tree height 2 to 8 for which incremental indexing proves to yield superior performance for a given DBMS. As the tree height grows, incremental indexing threshold values will vary. The incremental costs of sorting data need to be balanced against the cost of traversing the tree. Since both sorting and scanning the binary tree vary as $N\log N$ operations, to some extent the way these two factors trade off with varying amounts of new and old data depends on the implementation of the key sorting and tree insertion logic. Due to the dependency on implementation, the preferred embodiment models these costs empirically. Tests are performed to examine at what volumes of new index key values, incremental indexing performed better than full index rebuilding.

In the preferred embodiment, the $F(S_i)$ for the DB2™ product was determined by using as benchmark data a set of data provided by the Transactional Processing Performance Counsel. Based on that benchmark data, the values for the thresholds for tree heights 2 to 8, inclusive, for the DB2™ product were calculated as:

$T2 = 16$

$T3 = 19$

$T4 = 21$

14

CA9-99-022

T5 = 24

T6 = 27

T7 = 30

T8 = 33

5

Additional values may be calculated for tree heights greater than eight, as required. The above values indicate that where index 14 has a height 6, for example, and the data to be added to table 12 is above 27% of the size of table 12 (the number of records to be added is more than 27% of the number of records in table 12), then a full rebuild of index 12 will be more efficient than adding key

10 values to index 12 incrementally. It is to be understood that the above values may vary for different DBMS environments and data sets.

Although the above table values for the function $F(S_i)$ were arrived at by taking test data and manually constructing indexes to determine the thresholds for the different heights of trees, such a

15 process need not be carried out by a user. Once the location of the test data is indicated to the system of the preferred embodiment, it is possible for the system to run the tests to come up with the appropriate threshold values as part of the system set up. It is necessary for the system to clock its own processes and to select the appropriate threshold points based on the data provided.

20 For indexes such as hash indexes which are not based on a binary tree structure, the heuristic determination function may compare percentages of new data to total data volume to achieve a heuristic to predict when incremental or full rebuild approaches are more efficient.

Other characteristics of database 10 which may be included in determining the function $F(S_i)$

25 include:

     1. The percentage of freespace in index 14;

     2. The estimated size of index 14 after either incremental or full rebuild updates (a full

15

CA9-99-022

T5 = 24

T6 = 27

T7 = 30

T8 = 33

5

Additional values may be calculated for tree heights greater than eight, as required. The above values indicate that where index 14 has a height 6, for example, and the data to be added to table 12 is above 27% of the size of table 12 (the number of records to be added is more than 27% of the number of records in table 12), then a full rebuild of index 12 will be more efficient than adding key

10   values to index 12 incrementally. It is to be understood that the above values may vary for different DBMS environments and data sets.

Although the above table values for the function $F(S_i)$ were arrived at by taking test data and manually constructing indexes to determine the thresholds for the different heights of trees, such a

15   process need not be carried out by a user. Once the location of the test data is indicated to the system of the preferred embodiment, it is possible for the system to run the tests to come up with the appropriate threshold values as part of the system set up. It is necessary for the system to clock its own processes and to select the appropriate threshold points based on the data provided.

20   For indexes such as hash indexes which are not based on a binary tree structure, the heuristic determination function may compare percentages of new data to total data volume to achieve a heuristic to predict when incremental or full rebuild approaches are more efficient.

Other characteristics of database 10 which may be included in determining the function $F(S_i)$

25   include:

1. The percentage of freespace in index 14;

2. The estimated size of index 14 after either incremental or full rebuild updates (a full

15

CA9-99-022

rebuild will often reduce fragmentation);

     3. The width of the average index key;

     4. The size of the sort heap and the database bufferpool (for memory caching);

     5. The number and speed of the CPUs in the system; and

5   6. The number and speed of disks in the system.

Each of the above characteristics may have an impact on the relative efficiency of incremental and full rebuild updating of index 14. For that reason, inclusion of these values in the function $F(S_i)$ may provide additional predictive accuracy for the function.

10

Although a preferred embodiment of the present invention has been described here in detail, it will be appreciated by those skilled in the art, that variations may be made thereto, without departing from the spirit of the invention or the scope of the appended claims.

16

CA9-99-022

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1.   A computer system for database management comprising, means for storing and updating a
5   first set of data,

        indexing means for storing and updating a selected index of key values related to the first set of data, the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index,

        heuristic determination means for selecting the incremental update of the index, or the full
10   rebuild of the index, for a given second set of data to be added to the first set of data.

2.   The computer system of claim 1 in which the heuristic determination means comprises a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

15

3.   The computer system of claim 2 in which the database management system is a relational database management system in which the index is stored as a binary tree and in which the index meta-data comprises estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the function is empirically defined to select the
20   incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index.

4.   The computer system of claim 3 in which the function determines, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size
25   of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

17

CA9-99-022

5. The computer system of claim 1 in which the heuristic determination means comprises means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

5   6. The computer system of claim 2 further comprising a means for storing the index meta-data independent of the means for storing and updating the first set of data.

7. The computer system of claim 2 in which the means for storing the index meta-data comprises a recovery history file for the first set of data.

10

8. The computer system of claim 1 in which the selected index of key values is one of a plurality of indexes and in which the selected index is the first index on the first set of data.

9. The computer system of claim 1 in which the selected index of key values is one of a plurality
15   of indexes and in which the selected index is selected on the basis of the relative sizes of each of the plurality of indexes on the first set of data.

10. The computer system of claim 2 further comprising means for generating values for the function of the heuristic determination means, for a specified test range of values for each of the first
20   set of data, the index, and the second set of data.

11. The computer system of claim 2, the computer system having one or more CPUs, one or more disks, a sort heap and a database bufferpool, and in which the index meta-data is defined to reflect a subset of the following characteristics: the percentage of free space in the index, the estimated size
25   of the index after both the incremental and the rebuild updates of the index, the width of the average key value in the index, the size of the sort heap and the database bufferpool in the computer system, the number and speed of the CPUs in the computer system, and the number and speed of the disks

18

CA9-99-022

in the system.

12. A computer system for relational database management comprising,

means for storing and updating a first set of data,

5   indexing means for storing as a binary tree, and updating, a selected index of key values related to the first set of data the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index,

heuristic determination means for selecting the incremental update of the index, or the full

10   rebuild of the index, for a given second set of data to be added to the first set of data, the heuristic determination means comprising a function which takes as input index meta-data,

the index meta-data being stored separately from the first set of data and comprising estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index,

15   whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index,

the heuristic determination means furthercomprising means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

20

13. A computer program product for use with a computer comprising a central processing unit and random access memory, said computer program product comprising a computer usable medium having computer readable code means embodied in said medium for managing a database, said computer program product comprising:

25   computer readable program code means for causing a computer to store and update a first set of data,

computer readable program code indexing means for causing a computer to store and update

19

CA9-99-022

a selected index of key values related to the first set of data, the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index,

5   computer readable program code heuristic determination means for causing a computer to select the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data.

14.   The computer program product of claim 13 in which the heuristic determination means comprises a function which takes as input index meta-data, comprising characteristics of the first set
10   of data, the index, and the second set of data.

15.   The computer program product of claim 14 in which the database is a relational database in which the index is stored as a binary tree and in which the index meta-data comprises estimates of the table size of the first set of data, the table size of the second set of data, and the height of the
15   index, whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index.

16.   The computer program product of claim 15 in which the function determines, for each
20   potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

17.   The computer program product of claim 13 in which the heuristic determination means
25   comprises means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

20

CA9-99-022

18. A method for updating a selected index in a computer system for database management, the computer system comprising means for storing and updating a first set of data related to the index, the method comprising the steps of:

    (a) heuristically selecting one of: the incremental update of the index, or the full rebuild of

5   the index, for a given second set of data to be added to the first set of data, and

    (b) selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, as selected.

19. The method of claim 18 in which the step of heuristically selecting the update of the index

10   comprises the evaluation of a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

20. A method for updating a selected b-tree index in a computer system for relational database management, the computer system comprising means for storing and updating a first set of data

15   related to the index, the method comprising the steps of:

    (a) using a heuristic determination function to select one of: the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, the heuristic determination function taking as input index meta-data comprising estimates of the table size of the first set of data, the table size of the second set of data, and the height of the

20   index, whereby the heuristic function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index, and

    (b) selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, as selected.

25

21. The method of claim 20 in which the function comprises the step of determining, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of

CA9-99-022

data to the table size of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

22.   The method of claim 18 further comprising the step of a user specifying the selection of the incremental update of the index, the full rebuild of the index, or the selection based on the heuristic determination function.

23.   The method of claim 19 further comprising the step of storing the index meta-data independent of the means for storing and updating the first set of data.

24.   A computer program product tangibly embodying a program of instructions executable by a computer to perform the method steps of claim 18.

25.   A computer program product tangibly embodying a program of instructions executable by a computer to perform the method steps of claim 20.

26.   A computer program product tangibly embodying a program of instructions executable by a computer to perform the method steps of claim 21.
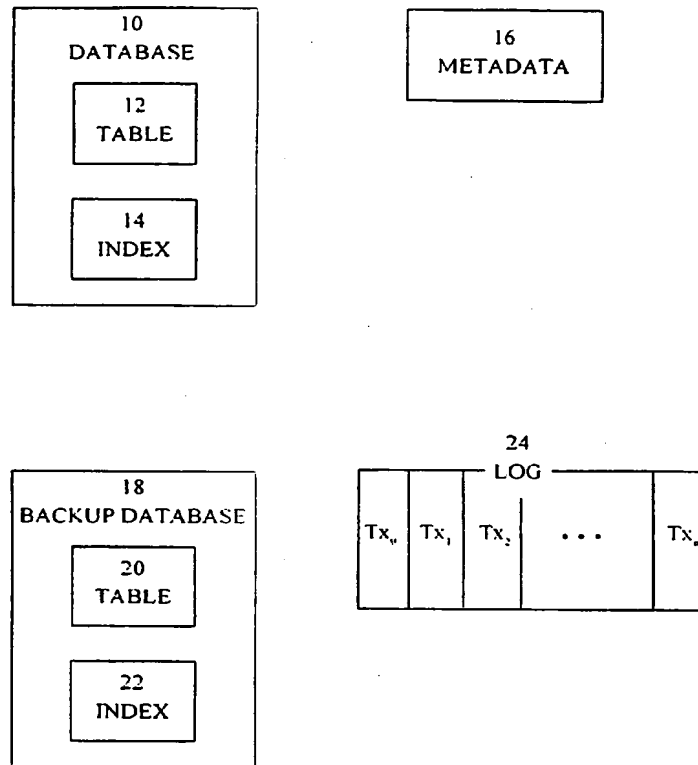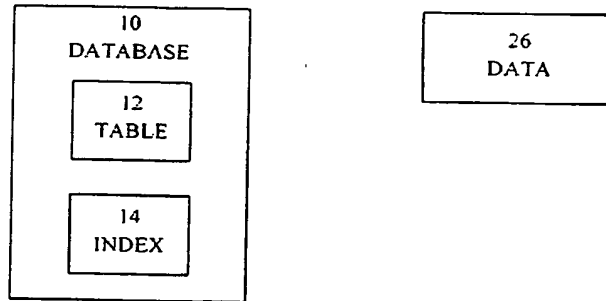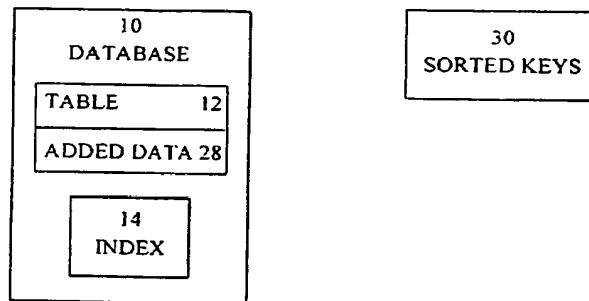
22

FIGURE 1

```
┌─────────────────────────┐              ┌──────────────────────┐
│           10            │              │         26           │
│        DATABASE         │              │        DATA          │
│   ┌─────────────────┐   │              └──────────────────────┘
│   │       12        │   │
│   │     TABLE       │   │
│   └─────────────────┘   │
│                         │
│   ┌─────────────────┐   │
│   │       14        │   │
│   │     INDEX       │   │
│   └─────────────────┘   │
└─────────────────────────┘
```

FIGURE 2

```
┌─────────────────────────┐              ┌──────────────────────┐
│           10            │              │         30           │
│        DATABASE         │              │    SORTED KEYS       │
│  ┌───────────────────┐  │              └──────────────────────┘
│  │  TABLE        12  │  │
│  ├───────────────────┤  │
│  │  ADDED DATA 28    │  │
│  └───────────────────┘  │
│                         │
│   ┌─────────────────┐   │
│   │       14        │   │
│   │     INDEX       │   │
│   └─────────────────┘   │
└─────────────────────────┘
```

FIGURE 3